

# Introduction to the Abstract Meaning Representation (AMR)

<http://tiny.cc/amrtutorial>

If you are here early, go to the **AMR Editor** and try to log in:

<http://tiny.cc/amreditor>

# Why does AMR matter now?

- AMR is a semantic representation aimed at **large-scale human annotation** in order to **build a giant semantics bank**.
- We do a practical, replicable amount of abstraction (**limited canonicalization**).
- Capture many aspects of meaning in **a single simple data structure**.

# Hasn't this been done before?

- Linguistics/CL have formalized semantics for a long time.
- A form of AMR has been around for a long time too (Langkilde and Knight 1998).
- It changed a lot since 1998 (add PropBank, etc.) and **we actually built a corpus of AMRs.**

# Contemporary AMR

- **Banarescu et al. (2013)** laid out the fundamentals of the annotation scheme we'll describe today.



# Roadmap for Part I

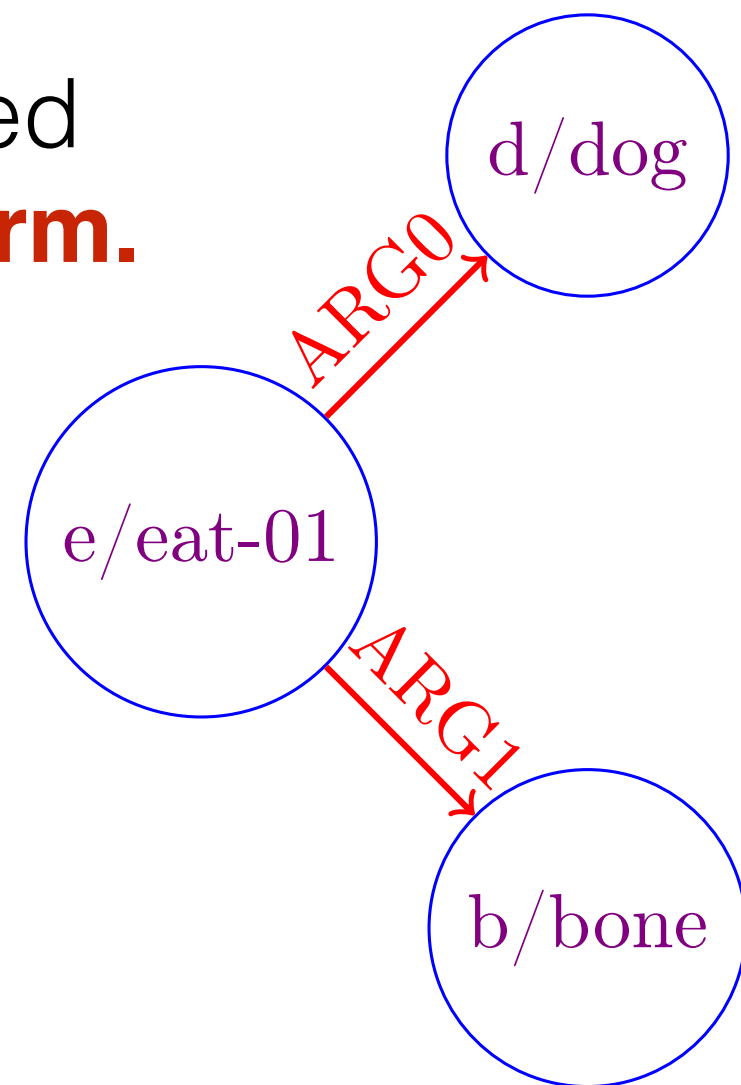
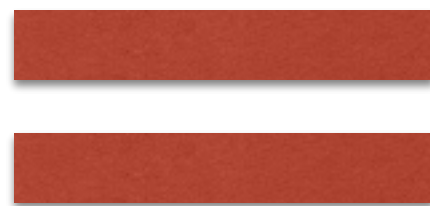
- Fundamentals of the AMR representation
- Hands-on practice I: Representing basic examples
- Advanced topics and linguistic phenomena
- Comparison to other representations
- Hands-on practice II: Doing real, complex text

# PENMAN notation

- We use PENMAN notation (Bateman 1990).
- A way of representing a directed graph in a **simple, tree-like form**.

*“The dog is eating a bone”*

(e / eat-01  
:ARG0 (d / dog)  
:ARG1 (b / bone))



# PENMAN notation

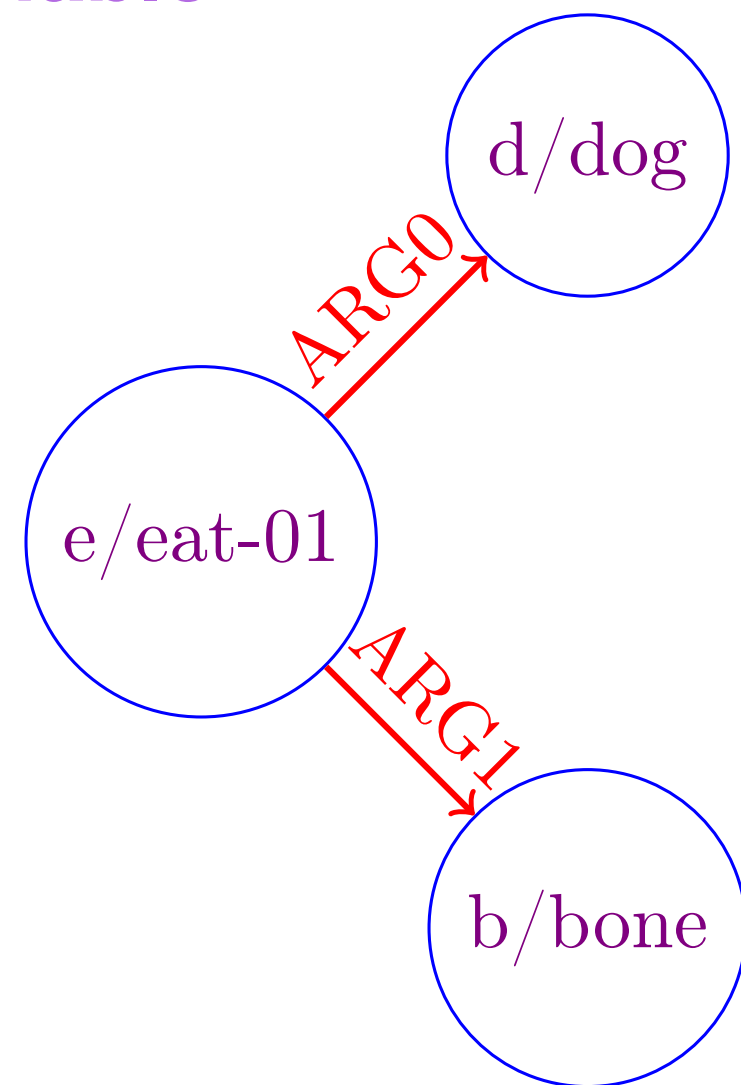
- The edges (ARG0 and ARG1) are **relations**
- Each node in the graph has a **variable**
- They are labeled with **concepts**
- **d** / **dog** means “**d** is an instance of **dog**”

*“The dog is eating a bone”*

(**e** / **eat-01**

:**ARG0** (**d** / **dog**)

:**ARG1** (**b** / **bone**))



# PENMAN notation

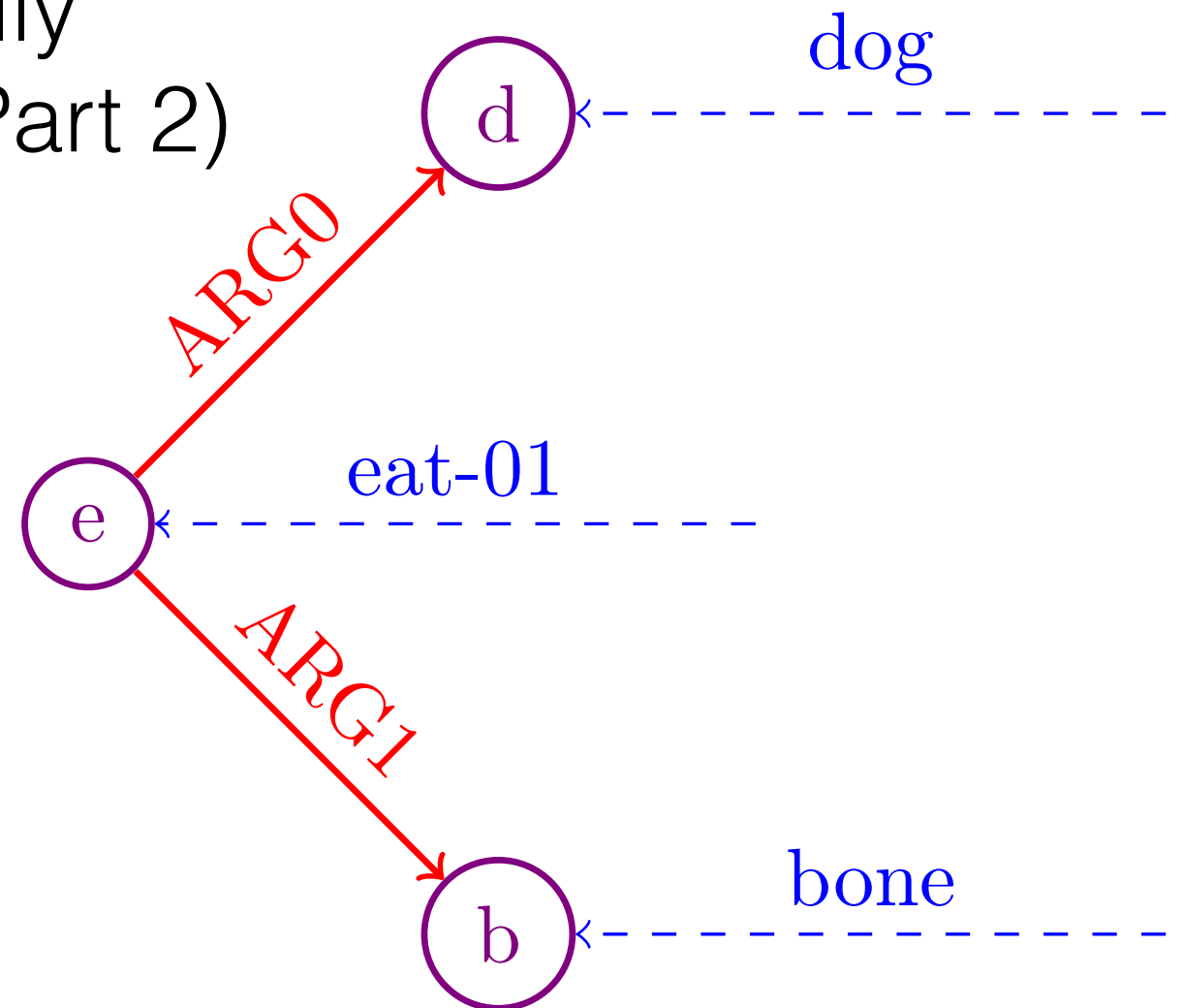
- **Concepts** are technically **edges** (this matters in Part 2)

*“The dog is eating a bone”*

(**e** / **eat-01**

:**ARG0** (**d** / **dog**)

:**ARG1** (**b** / **bone**))



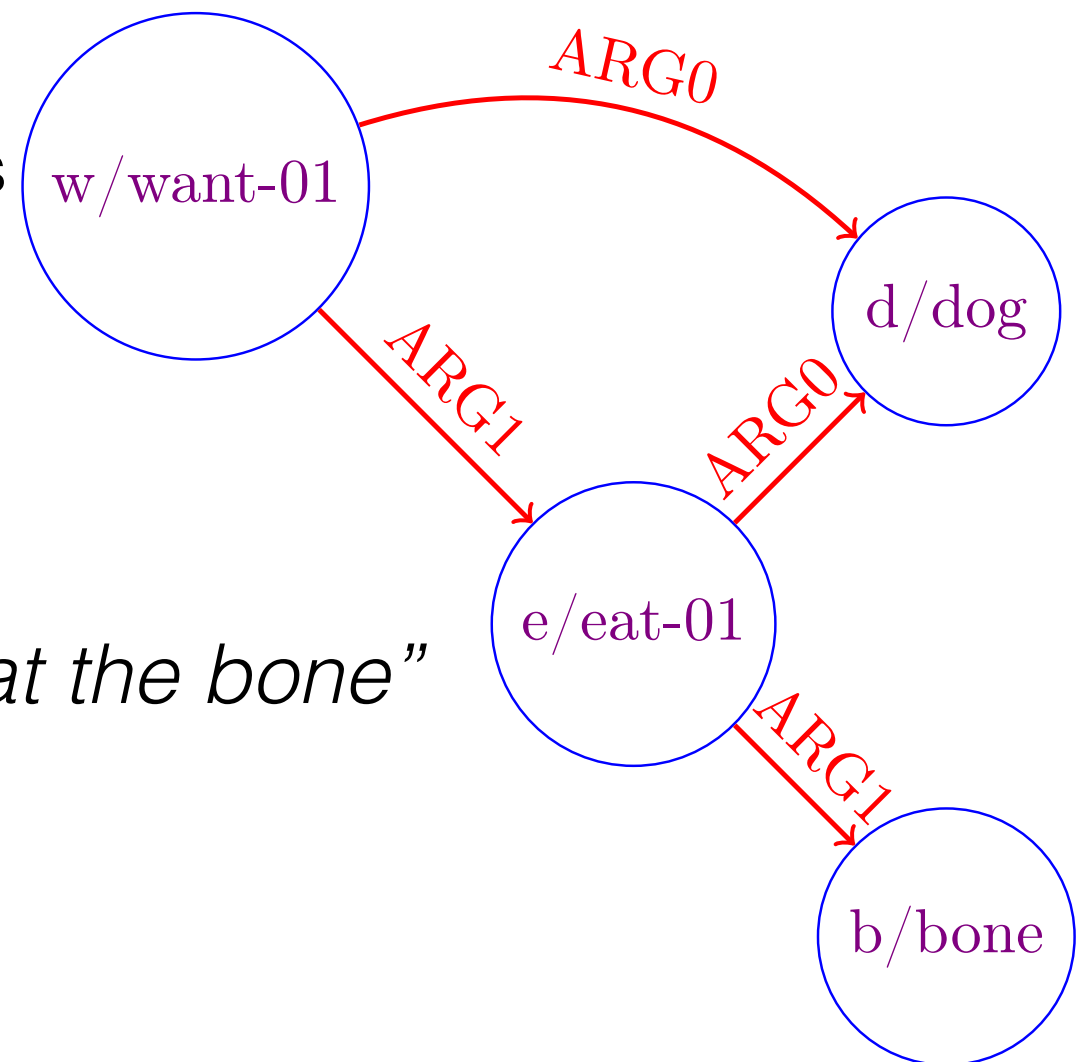


# Reentrancy

- What if something is referenced multiple times?
- Notice how **dog** has two incoming roles now.
- To do this in PENMAN format, repeat the variable. We call this a **reentrancy**.

*“The dog **wants to** eat the bone”*

```
(want-01
:ARG0 (d / dog)
:ARG1 (e / eat-01
:ARG0 d
:ARG1 (b / bone)))
```



# Reentrancy

- It **does not matter** where the concept label goes.

*“The dog wants to eat the bone”*

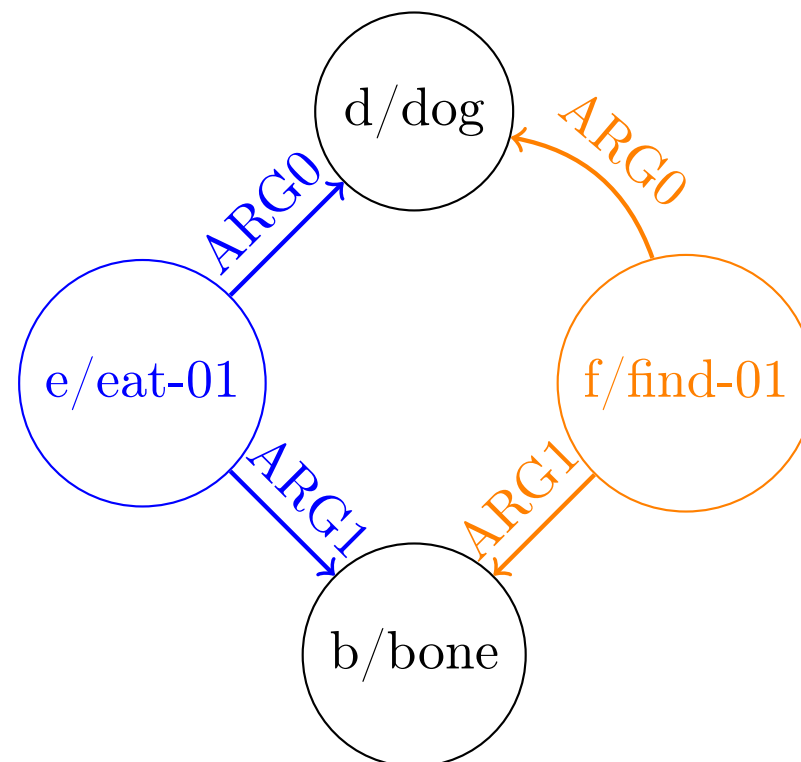
(want-01  
:ARG0 (**d / dog**)  
:ARG1 (e /eat-01  
:ARG0 **d**  
:ARG1 (b / bone)))



(want-01  
:ARG0 **d**  
:ARG1 (e /eat-01  
:ARG0 (**d / dog**)  
:ARG1 (b / bone)))

# Inverse Relations and Focus

- What about “The dog ate **the bone that he found**”?



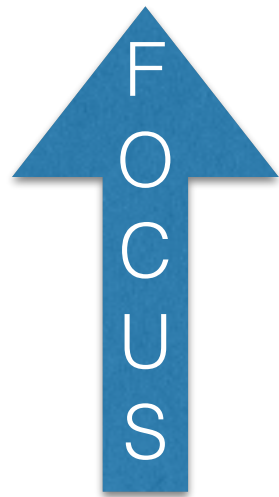
- How do we know **what goes on top**?
- How do we get these **into the AMR format**?

# Inverse Relations and Focus

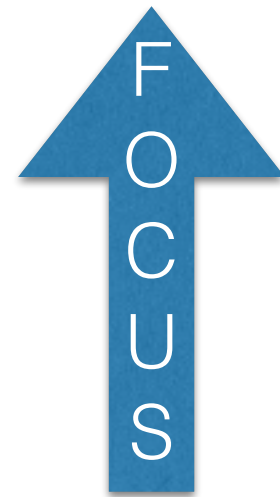
- We call “what goes on top” the **focus**.
- Conceptually, **the main assertion**.
- Linguistically, **often the head**.
  - ▶ For a sentence, **usually the main verb**.

# Inverse Relations and Focus

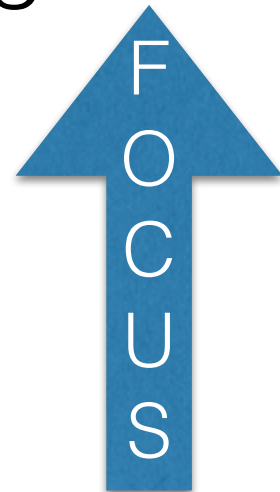
The man at the hotel



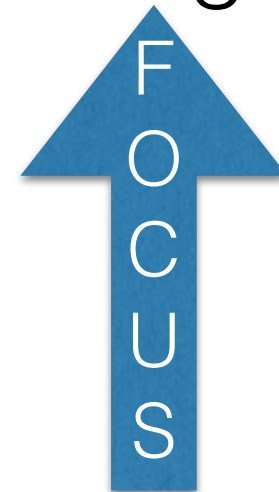
The hotel the man is at



The dog ran

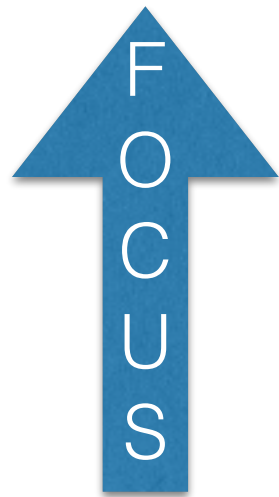


The dog that ran



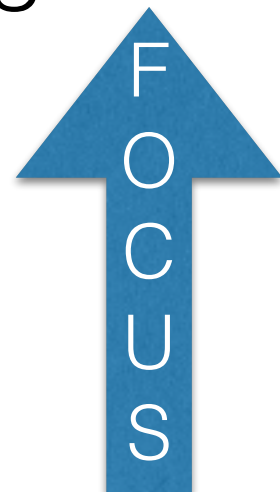
# Inverse Relations and Focus

The man at the hotel



(m / man  
**:location** (h / hotel))

The dog ran

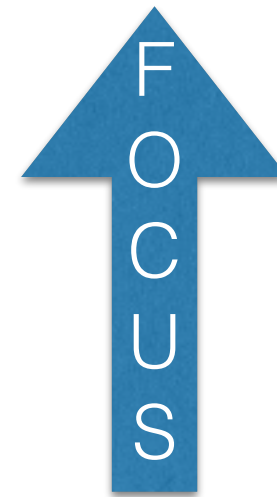


(r / ran-01  
**:ARG0** (d / dog))

# Inverse Relations and Focus

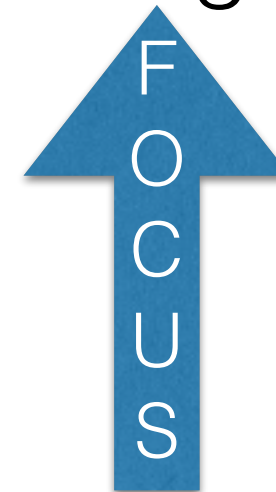
(h / hotel  
:???) (m / man))

The hotel the man is at



(d / dog  
:?????) (r / ran-01))

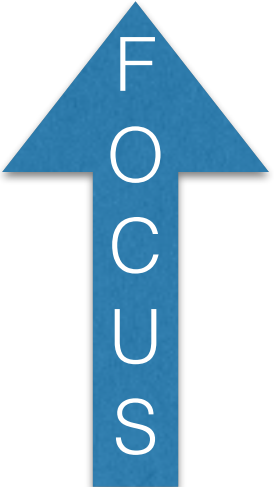
The dog that ran



# Inverse Relations and Focus

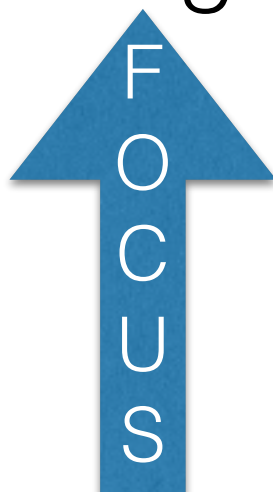
The hotel the man is at

(h / hotel  
:location-of (m / man))



The dog that ran

(d / dog  
:ARG0-of (r / ran-01))



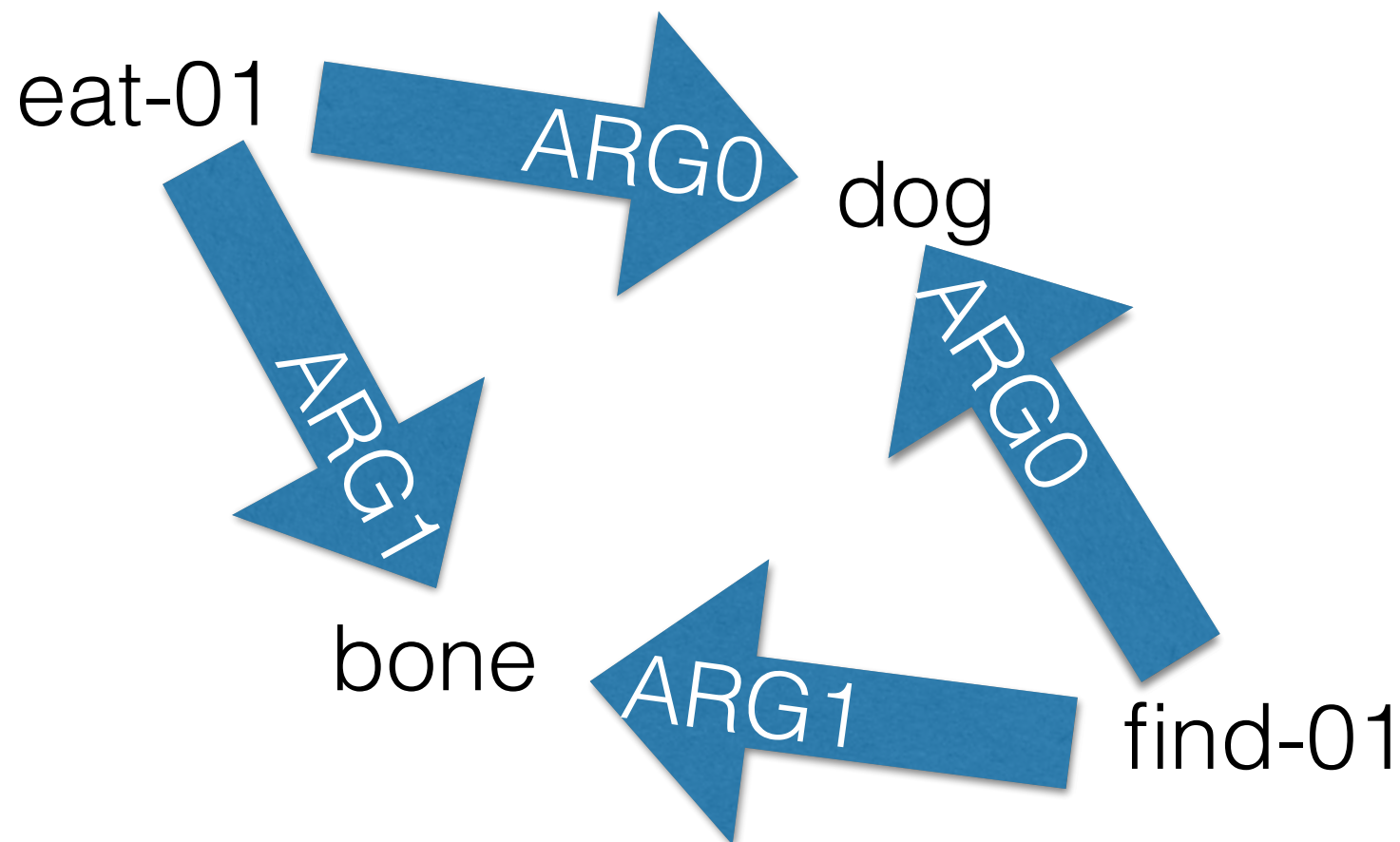


# Inverse Relations and Focus

- This is a notational trick: **X ARG0-of Y = Y ARG0 X**
- Often used for relative clauses.
- *These are equivalent for SMATCH scoring purposes too.*

# Reviewing the Format

- Imagine a graph for “***The dog ate the bone that he found***”

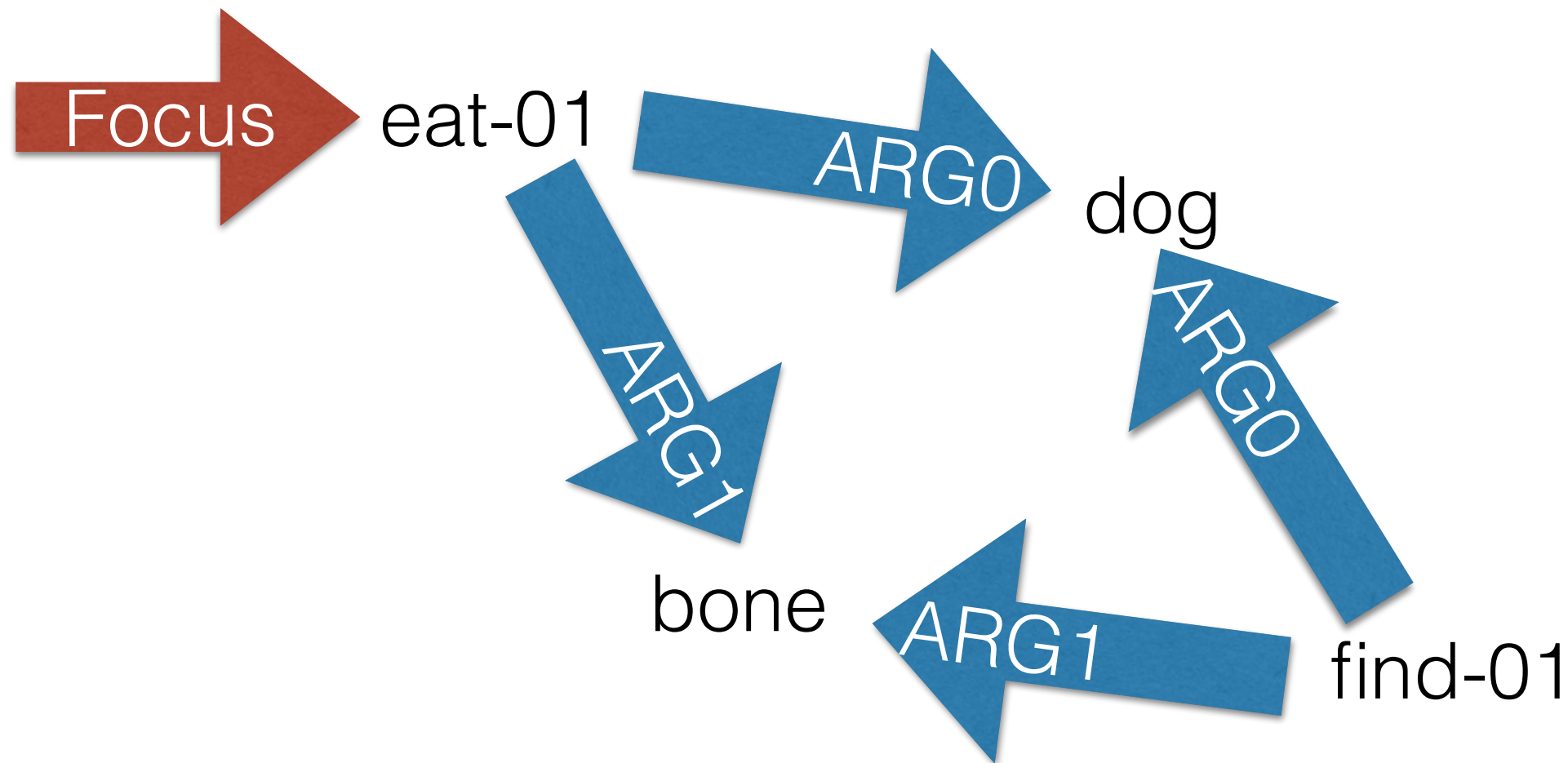


- “***The dog ate the bone that he found***”

# Reviewing the Format

(e / eat-01 ...)

- Find the focus

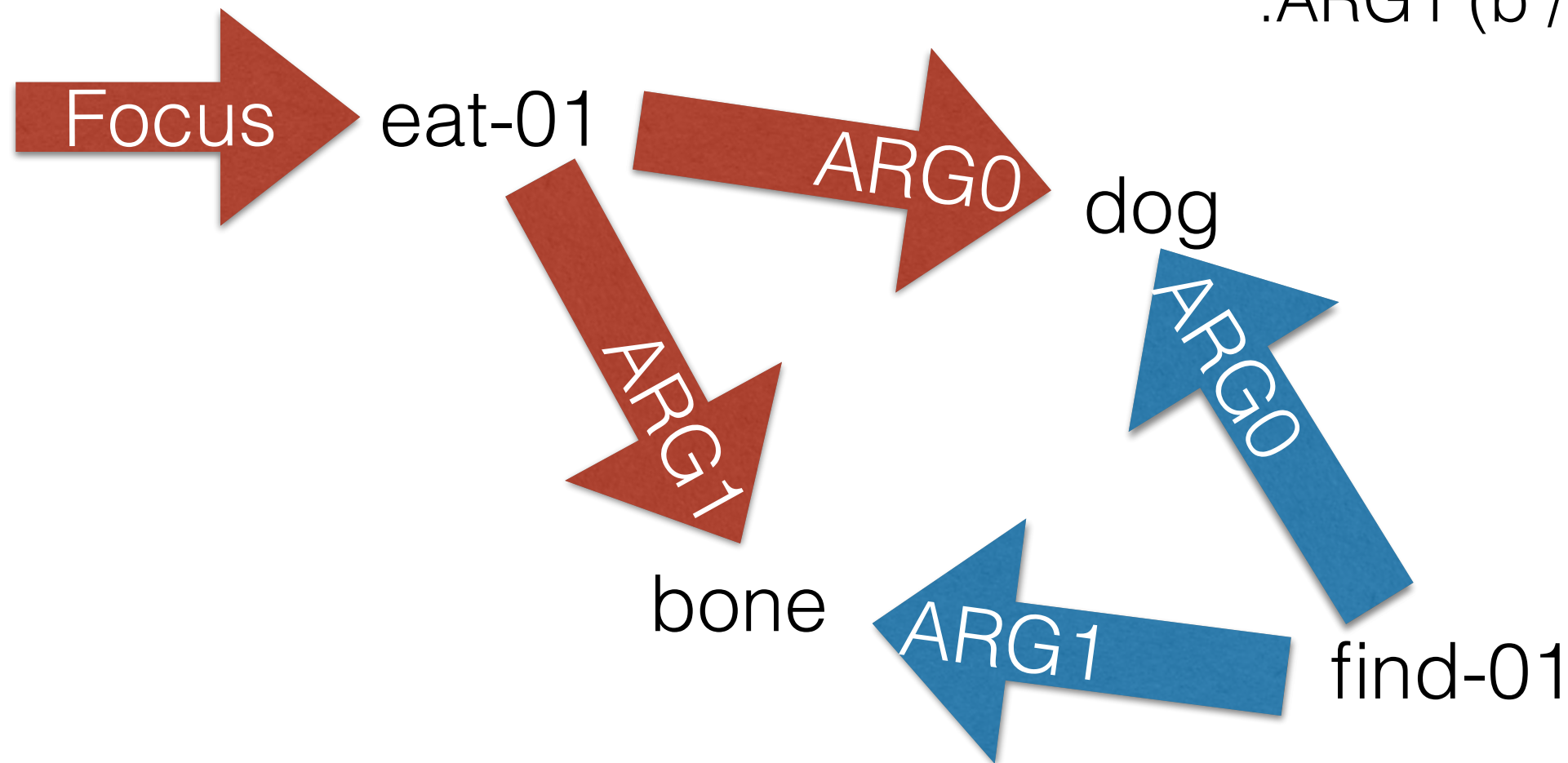


- ***"The dog ate the bone that he found"***

# Reviewing the Format

- Add entities

(e / eat-01  
:ARG0 (d / dog)  
:ARG1 (b / bone))



- “*The dog ate the bone that he found*”

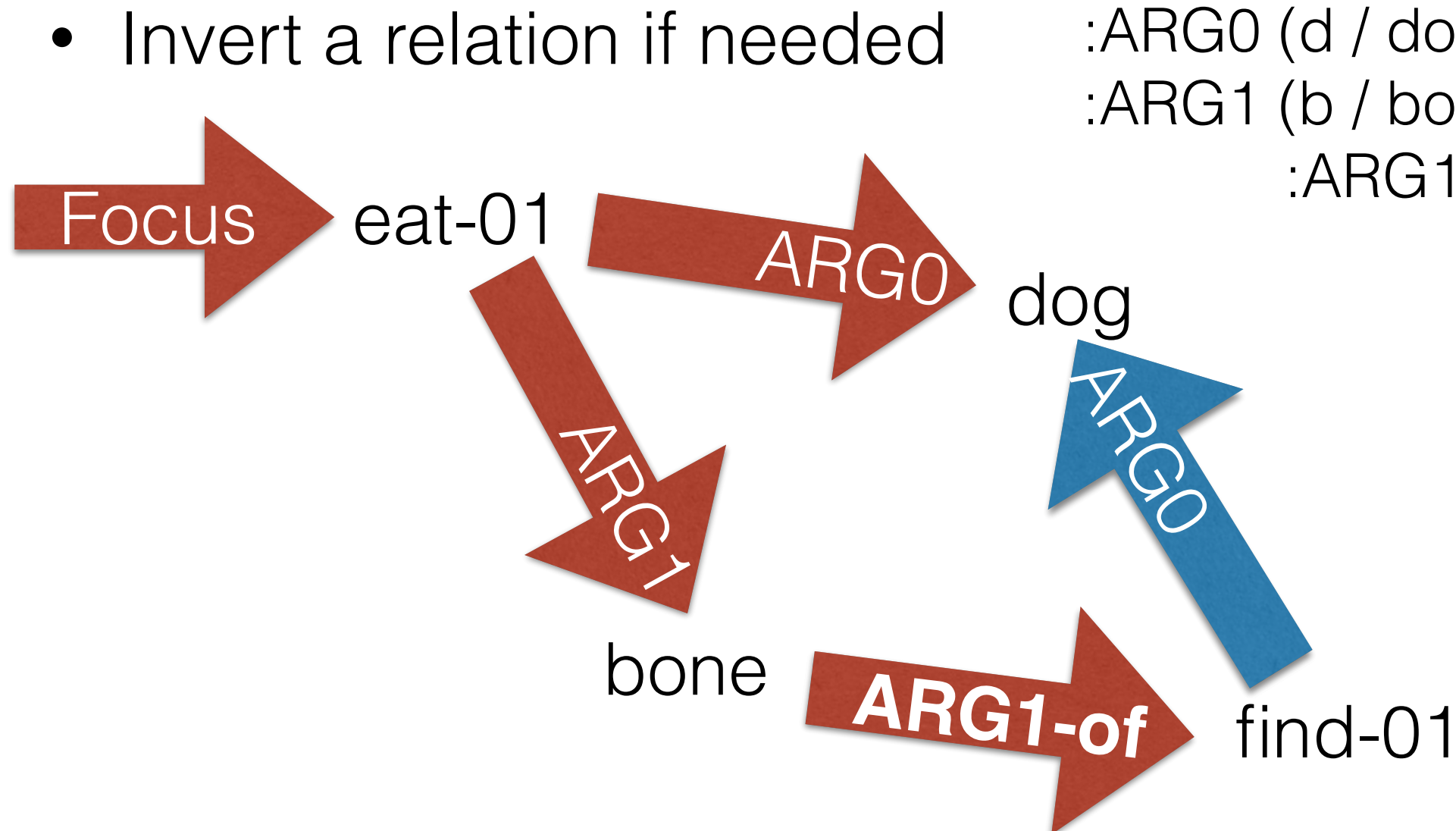
# Reviewing the Format

(e / eat-01

:ARG0 (d / dog)

:ARG1 (b / bone

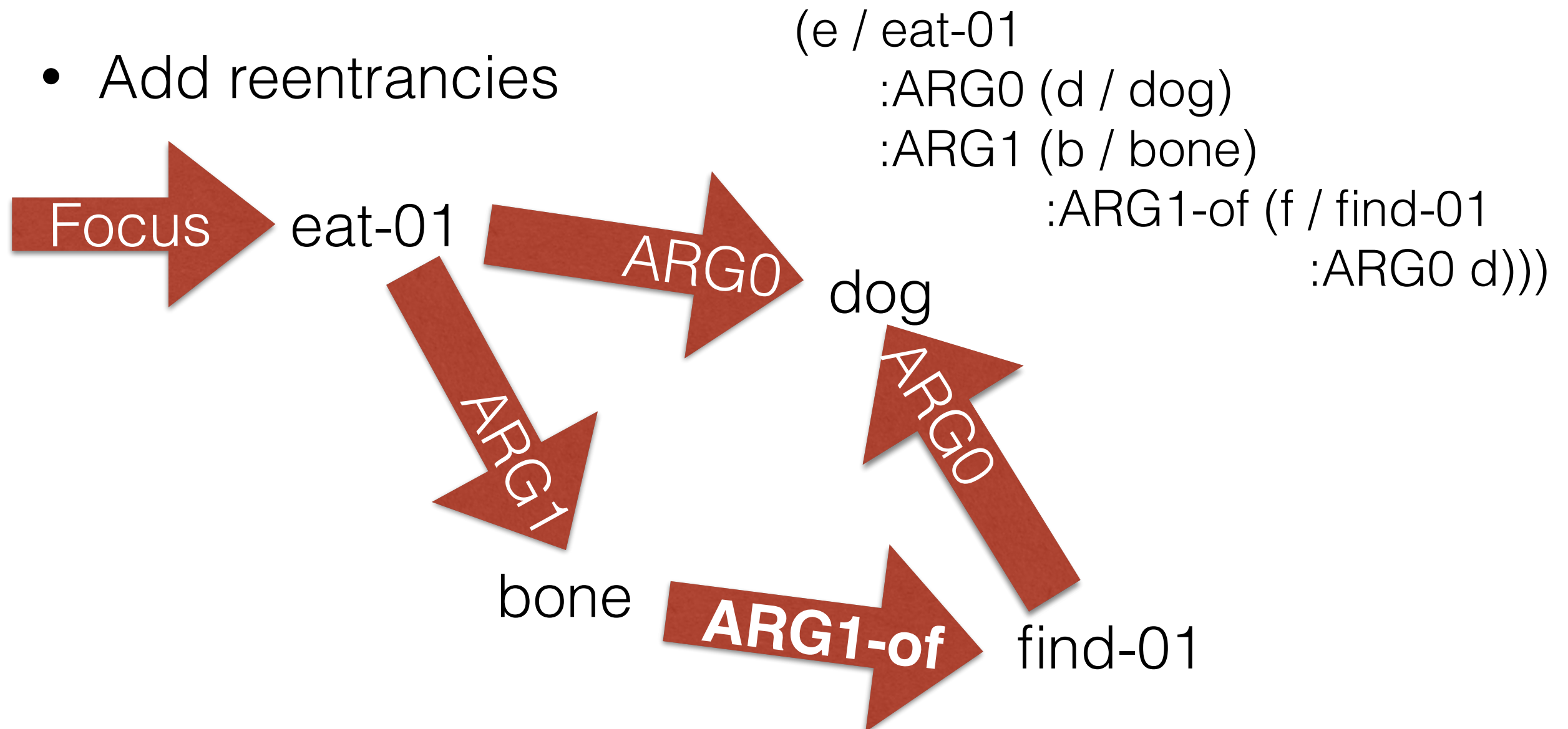
:ARG1-of (f / find-01)))



- “*The dog ate the bone that he found*”

# Reviewing the Format

- Add reentrancies



- ***“The dog ate the bone that he found”***

# Constant

- Some relations, called **constants**, get no **variable**.

- The editor does this **automatically** for certain contexts.

- This happens for **negation**.

*“The dog **did not** eat the bone”*  
(e /eat-01 **:polarity -**  
:ARG0 (d / dog)  
:ARG1 (b / bone))

# Constant

- Some relations, called **constants**, get no **variable**.
- The editor does this **automatically** for certain contexts.

*“The dog ate **four** bones”*  
(e /eat-01  
:ARG0 (d / dog)  
:ARG1 (b / bone **:quant 4**))
- This happens for **numbers**.

*(to create a concept starting with a nonalphabetic character, type “!” before the concept)*



# Constant

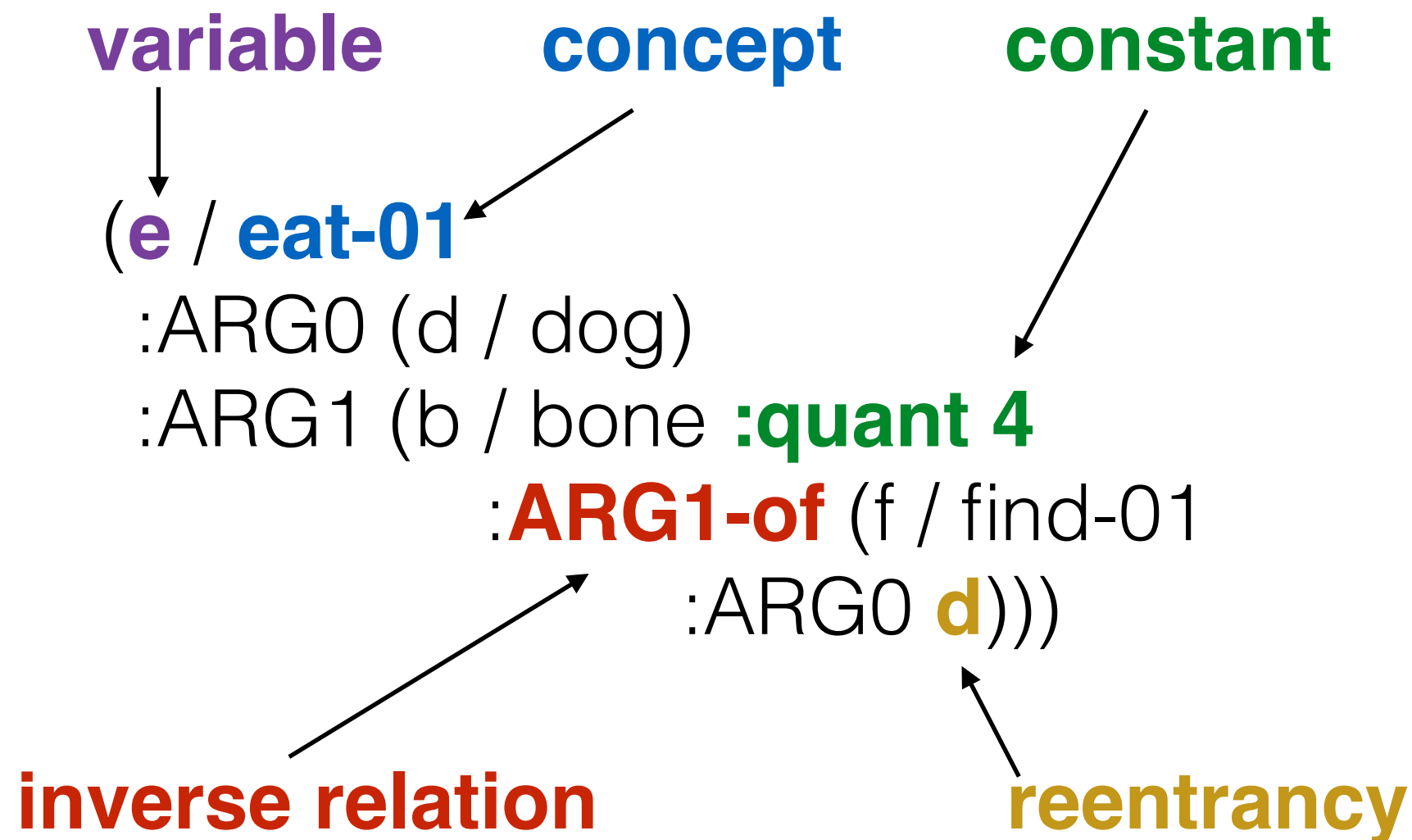
- Some relations, called **constants**, get no **variable**.
- The editor does this **automatically** for certain contexts.

***Fido** the dog*  
(d / dog  
:name (n / name :op1 **"Fido"**))
- This happens for **names**

# Concepts vs. Constants

- A **concept** is a type. For every concept node there will be  $\geq 1$  instance variable/node.
  - An instance can be mentioned multiple times.
  - Multiple instances of the same concept can be mentioned.
- **Constants** are singleton nodes: no variable, just a value. Specific non-core roles allow constant values.

- That's AMR notation! Let's review before discussing **how we annotate AMRs**.



# PropBank Lexicon

- Predicates use the ***PropBank*** inventory.
- Each frame presents annotators with a list of senses.
- Each sense has its own definitions for its numbered (core) arguments

## run-01 - "operate, proceed, operate or proceed"

- ARG0: operator
- ARG1: machine, operation, procedure
- ARG2: employer
- ARG3: coworker
- ARG4: instrumental

Aliases: [run](#) (v), [run](#) (n), [running](#) (n)  
[more](#)

## run-02 - "walk quickly, a course or contest, run/jog, run for office"

- ARG0: runner **theme**
- ARG1: course, race, distance **location**
- ARG2: opponent

Aliases: [run](#) (v), [run](#) (n), [running](#) (n)  
[more](#)

## run-03 - "cost"

- ARG1: commodity
- ARG2: price
- ARG3: buyer

Aliases: [run](#) (v), [running](#) (n)

# PropBank Lexicon

- We generalize across **parts of speech** and **etymologically related words**:

<i>My fear of snakes</i>	fear-01
<i>I am fearful of snakes</i>	fear-01
<i>I fear snakes</i>	fear-01
<i>I'm afraid of snakes</i>	fear-01

- But we **don't** generalize over synonyms:

<i>My fear of snakes</i>	fear-01
<i>I'm terrified of snakes</i>	terrify-01
<i>Snakes creep me out</i>	creep_out-03

# Stemming Concepts

- Non-predicates don't have PropBank frames. They are simply stemmed.
- All concepts drop **plurality**, **articles**, and **tense**.

A cat  
The cat  
cats  
the cats

(c / cat)

eating  
eats  
ate  
will eat

(e / eat-01)

# Why drop articles?

- All mentions of a term go to **the same variable**, including **pronouns** and **later nominal mentions**.

*I saw **a nice dog** and noticed **he** was eating a bone*



(d / dog  
:mod nice)

*Is “d” indefinite  
or definite?*

- We do capture **demonstratives**:

*This house*

(h / house  
:mod (t / this))

# Stemming Concepts

- Pronouns that do not have a coreferent nominal mention are **made nominative** and **used as normal concepts**.

The man saved himself    **He** saved himself    He saved **me**

(s / save-01  
:ARG0 (m / man)  
:ARG1 m)

(s / save-01  
:ARG0 (**h / he**)  
:ARG1 h)

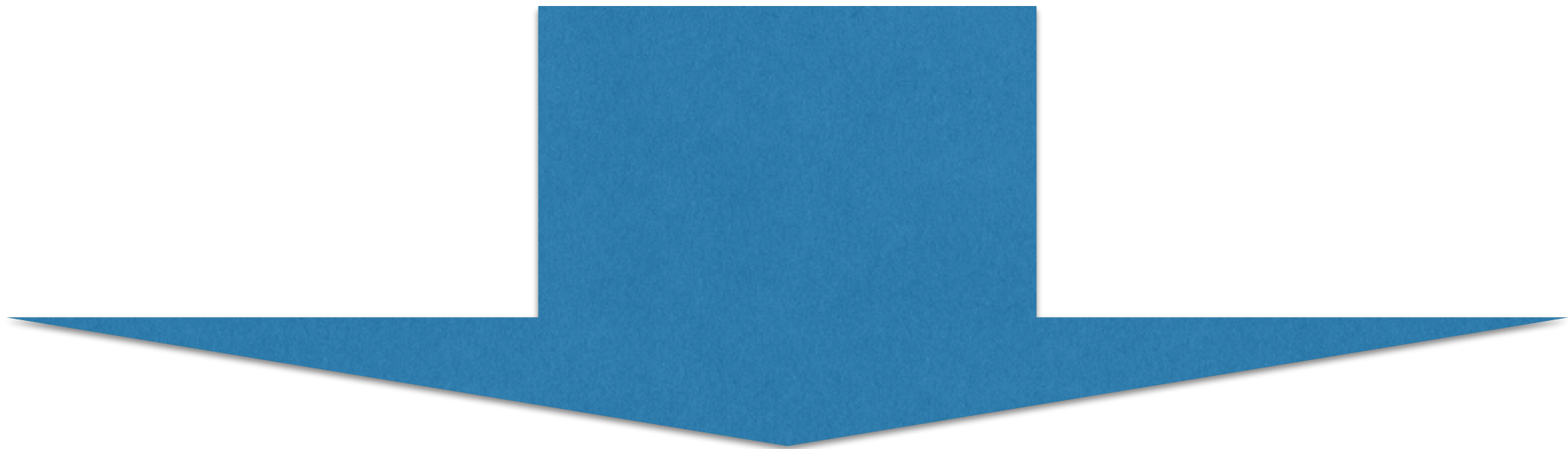
(s / save-01  
:ARG0 (h / he)  
:ARG1 (**i / i**))



# Why drop tense?

- English verbal tense **doesn't generalize well cross-linguistically**; not available for nominal predicates.
- Richer time representation might have required looking beyond a sentence.
- Keep a simple representation.

*The man described the mission as a disaster.*  
*The man's description of the mission: disaster.*  
*As the man described it, the mission was a disaster.*  
*The man described the mission as disastrous.*



(d / describe-01  
:ARG0 (m / man)  
:ARG1 (m2 / mission)  
:ARG2 (d / disaster))

# Non-core Role Inventory

- If a semantic role is not in the **core roles** for a roleset, AMR provides an inventory of **non-core roles**

## run-01 - "operate, proceed, operate or proceed"

- ARG0: operator
- ARG1: machine, operation, procedure
- ARG2: employer
- ARG3: coworker
- ARG4: instrumental

- These express things like **:time**, **:manner**, **:part**, **:location**, **:frequency**
- Inventory on handout, or in editor (the **[roles]** button)

- **General semantic roles (incl. shortcuts):** [:accompanier ex](#) [:age ex](#) [:compared-to ex](#) [:concession ex](#) [:condition ex](#) [:consist-of ex](#) [:cos](#) [:direction ex](#) [:domain ex](#) [:duration ex](#) [:employed-by ex](#) [:example](#) [:instrument ex](#) [:li ex](#) [:location ex](#) [:manner ex](#) [:meaning ex](#) [:med](#) [:name ex](#) [:ord ex](#) [:part ex](#) [:path ex](#) [:polarity ex](#) [:polite ex](#) [:poss](#) [:source ex](#) [:subevent ex](#) [:subset ex](#) [:superset ex](#) [:time ex](#) [:topic ex](#)
- **In quantities:** [:quant ex](#) [:unit ex](#) [:scale ex](#) [examples](#) [quantity ty](#)
- **In date-entity:** [:day](#) [:month](#) [:year](#) [:weekday](#) [:time](#) [:timezone ex](#) [:year2](#) [:decade](#) [:century](#) [:calendar ex](#) [:era ex](#) [:mod](#) [date-entity ex](#)
- **Ops:** [:op1](#) [:op2](#) [:op3](#) [:op4](#) [:op5](#) [:op6](#) [:op7](#) [:op8](#) [:op9](#) [:op10](#)
- **In multi-sentence:** [:snt1](#) [:snt2](#) [:snt3](#) [:snt4](#) [:snt5](#) [:snt6](#) [:snt7](#) [:snt8](#)

# Non-core Role Inventory

- We use **:mod** for attribution, and **:domain** is the inverse of mod (**:domain** = **:mod-of**)

*The yummy food*  
*There is yummy food*  
(f / food  
**:mod** (y / yummy))

*The yumminess of the food*  
*The food is yummy*  
(y / yummy  
**:domain** (f / food))

*seeing the yummy food*  
*seeing the food that is yummy*  
(s / see-01  
:ARG1 (f / food  
**:mod** (y / yummy)))

*seeing **that** the food is yummy*  
(s / see-01  
:ARG1 (y / yummy  
**:domain** (f / food)))

# Non-core Role Inventory

- This is also used for attribute/predicative demonstratives and nominals

*This house*

(h / house  
:mod (t / this))

*This is a house*

(t / this  
:domain (h / house))

*A monster truck*

(t / truck  
:mod (m / monster))

*the truck is a monster*

(m / monster  
:domain (t / truck))

# Non-core Roles: **:op#**

- Some relations need to have an ordered list of arguments, but **don't have specific meanings** for each entry.
- We use **:op1**, **:op2**, **:op3**, ... for these

# :op# for coordination

- We use this for coordination:
- *Apples and bananas* (a / and  
:op1 (a2 / apple)  
:op2 (b / banana))

# :op# for names


- *Barack Obama*  
(p / person  
:name (n / name  
:op1 "Barack"  
:op2 "Obama"))
- *Obama*  
(p / person  
:name (n / name  
:op1 "Obama"))



# Named Entities

- *Barack Obama*
- Entities with names get special treatment!
- We assign a **named entity type** from our ontology.
- 70+ categories like ***person, criminal-organization, newspaper, city, food-dish, conference***
- See your handout, or the **[NE types]** button in the editor

(**p / person**  
:name (n / name  
:op1 "Barack"  
:op2 "Obama"))



# Named Entities

- *Barack Obama* (p / person  
:name (n / name  
:op1 "Barack"  
:op2 "Obama"))
- Entities with names get special treatment!
- Each gets a :name relation to a name node
- That node gets :op# relations to the strings of their name *as used in the sentence.*

# Named Entities

- If there is a more specific descriptor present in the sentence, we use that **instead of the NE inventory.**
- *a Kleenex*  
(**p / product**  
:name (n / name  
:op1 "Kleenex"))
- *a Kleenex tissue*  
(**t / tissue**  
:name (n / name  
:op1 "Kleenex"))

# Wikification

- In a second pass of annotation, we add **:wiki** relations.

- *Barack Obama*

(p / person  
:name (n / name  
:op1 "Barack"  
:op2 "Obama")  
**:wiki Barack\_Obama**)



- [http://en.wikipedia.org/wiki/Barack\\_Obama](http://en.wikipedia.org/wiki/Barack_Obama)

# Measurable Entities

- We also have special entity types we use for **normalizable entities**.

*“Tuesday the 19th”*

*“five bucks”*

(**d / date-entity**

:weekday (t / tuesday)  
:day 19)

(**m / monetary-quantity**

:unit dollar  
:quant 5)

# Measurable Entities

- We also have special entity types we use for **normalizable entities**.

*“\$3 / gallon”*

(**r / rate-entity-91**

:ARG1 (m / monetary-quantity

:unit dollar

:quant 3)

:ARG2 (v / volume-quantity

:unit gallon

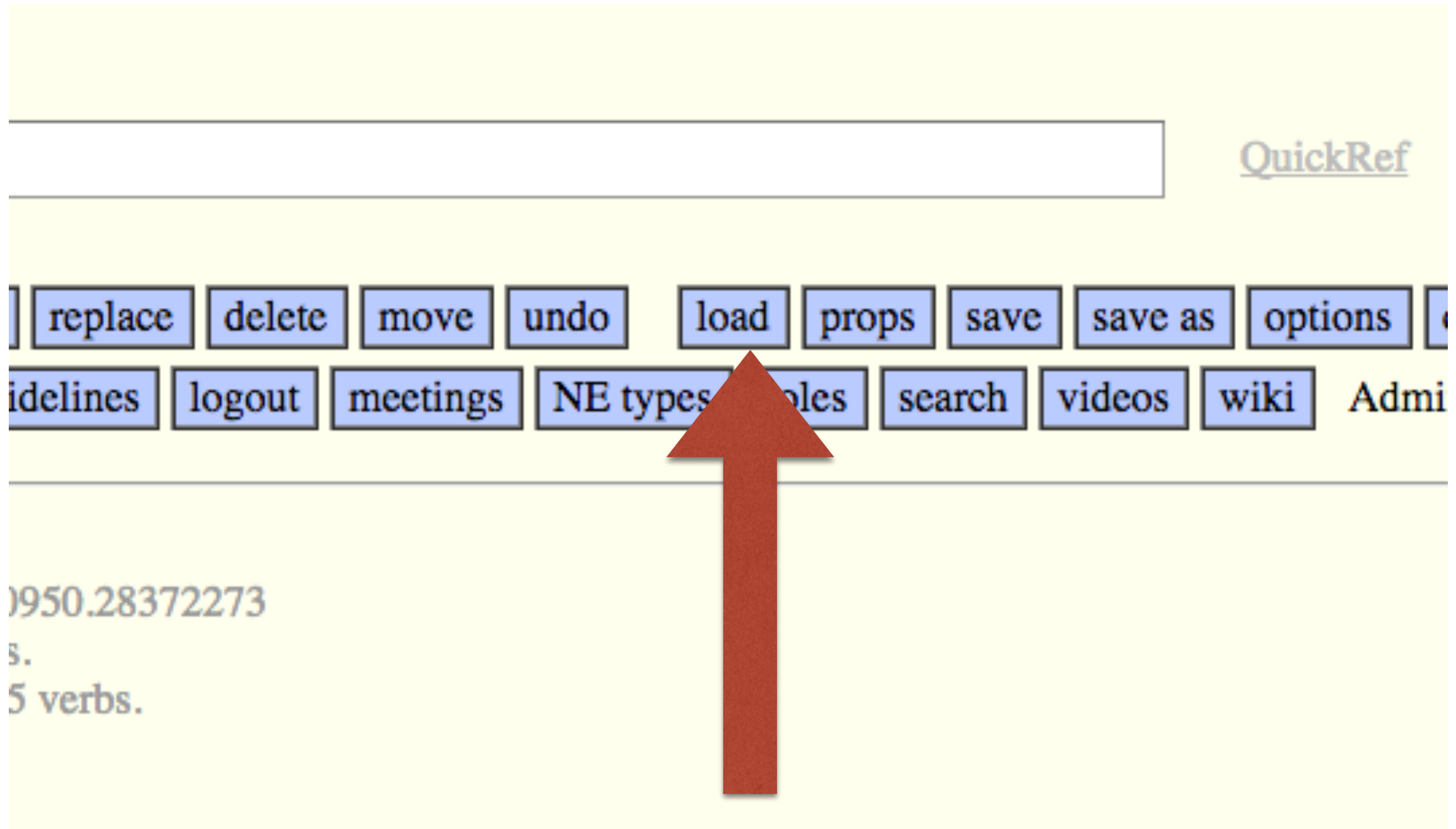
:quant 1))

# Hands-on Annotation!

Go to the **AMR Editor**:

<http://tiny.cc/amreditor>

# Load the Tutorial Sentences





# Select “NAACL Tutorial”

The screenshot shows a web interface with a menu bar at the top containing buttons: te, move, undo, load, props, save, save as, options, clear, help. Below this is a second row of buttons: meetings, NE types, roles, search, videos, wiki, Admin: AR, CA, CW, SW. The main area has a light blue background with several sections. The first section has a label 'an' followed by 'Workse' and a red arrow pointing to a 'Load workset at ISI' button. Below this is a text input field with the label '(without path):' followed by '.txt' and a 'Load file at ISI' button. The next section has a dropdown menu showing 'v1.7' with up/down arrows, followed by 'Snt. ID:' and another text input field, and a 'Load ON Sentence' button. The bottom section is partially visible, showing the letter 'I.'.

# It should look like this

**Sentence:** Tim likes to represent semantics abstractly

*empty AMR*

Enter text command:

Last command: Load next

Or select an action template:

[top](#) [add](#) [add-ne](#) [replace](#) [delete](#) [move](#) [undo](#) [exit](#)

**Workset NAACL-tutorial-set 1/59** [◀](#) col\_1008.1 (saved) [▶](#) Next: col\_1008.2

More:

[check](#) [copy](#) [dict](#) [diff](#) [generate](#) [guidelines](#) [logout](#) [meetings](#) [NE types](#)

**Log:** initialized empty AMR

for role checking: loaded 128 roles and 11 non-roles

# Commands

Use “top <concept>” to make a top node

**Sentence:** Tim likes to represent semantics abstractly

*empty AMR*

Enter text command:

Last command:      Load next

Or select an action template:

**Workset** NAACL-tutorial-set 1/59   ◀ col\_1008.1 (saved) ▶ Next: col\_1008.2  

More:

**Log:** initialized empty AMR  
For role checking, loaded 128 roles and 11 non-roles.  
For OntoNotes frame availability check, loaded 6245 verbs.

# Commands

Click on “like” to select the right sense

**Sentence:** Tim likes to represent

(1 / [like](#))

Enter text command:

Last command: top like

Or select an action template:

**Workset NAACL-tutorial**

More: [check](#) [copy](#) [dict](#)

about:blank

**Current sentence:** Tim likes to represent sen

**OntoNotes 4.0 frames**

Generated by Ulf's script on-frame-xml2html.pl on We

**Lemma: like (v)**

Note: Frames file for 'like' based on survey of

[like.01](#) - "affection"

# Commands

New relation: <variable> :<role> <concept>

**Sentence:** Tim likes to represent semantics abstractly

(1 / [like-01](#))

Enter text command:

Last command: del r

Or select an action template: [top](#) [add](#) [add-ne](#) [replace](#) [delete](#) [move](#) [undo](#) [exit/load](#) [props](#)

**Workset NAACL-tutorial-set 1/59** col\_1008.1 [Save and load next](#) [Discard and load next](#) Next: c

More: [check](#) [copy](#) [dict](#) [diff](#) [generate](#) [guidelines](#) [logout](#) [meetings](#) [NE types](#) [roles](#) [search](#)

**Log:** initialized empty AMR

For role checking, loaded 128 roles and 11 non-roles.

For OntoNotes frame availability check, loaded 6245 verbs.



# Commands

Anything after the third element is made into a name

Search documents and file names for text semantics abstractly

1 / [like-01](#)

:ARG1 (r / [representation-02](#)))

Enter text command: | :arg0 person Tim

Last command: replace concept at r with representation-02

Or select an action template: [top](#) [add](#) [add-ne](#) [replace](#) [delete](#) [move](#) [undo](#)

Workset NAACL-tutorial-set 1/59 col\_1008.1 [Save and load next](#) [Discard and](#)

More: [check](#) [copy](#) [dict](#) [diff](#) [generate](#) [guidelines](#) [logout](#) [meetings](#) [NE t](#)

# Commands

Make reentrancies with <variable> :<role> <variable>

**Sentence:** Tim likes to represent semantics abstractly

(1 / like-01  
:ARG0 (p / person :name (n / name :op1 "Tim"))  
:ARG1 (r / representation-02))

Enter text command:

Last command: 1 :arg0 person Tim

Or select an action template:

**Workset NAACL-tutorial-set 1/59** col\_1008.1

More:

Log initialized empty AMP

# Commands

When you are done, use “Save and Load Next”

**Sentence:** Tim likes to represent semantics abstractly

/ like-01

:ARG0 (p / person :name (n / name :op1 "Tim"))

:ARG1 (r / representation-02

:ARG0 p

:ARG1 (s / semantics)

:manner (a / abstract)))

Enter text command:

Last command: r :manner abstract

Or select an action template:

top

add

add-ne

replace

delete

move

undo

exit/load

props

Workset NAACL-tutorial-set 1/59

col\_1008.1

Save and load next

Discard and load next

Next: col\_1008.1

More:

check

copy

dict

diff

generate

guidelines

logout

meetings

NE types

roles

search



# Try the next sentence!

We will walk through it momentarily

**Sentence:** I hope Dumbledore likes my orange socks.

*empty AMR*

Enter text command:

Last command: Save and load next

Or select an action template:

[top](#)[add](#)[add-ne](#)[replace](#)[delete](#)[move](#)[undo](#)[exit/load](#)[pro](#)

**Workset NAACL-tutorial-set 2/59**



col\_1008.2 (saved)



Next: col\_1008.3

sent. me

More:

[check](#)[copy](#)[dict](#)[diff](#)[generate](#)[guidelines](#)[logout](#)[meetings](#)[NE types](#)[roles](#)[sea](#)

**Log:** initialized empty AMR

for role checking, loaded 128 roles and 11 non-roles.

for OntoNotes frame availability check, loaded 6245 verbs.